We claim:

1. A SIMD machine with an array comprising at least one processing element (PE) and an array control processor, the array control processor and the PE each comprising:

an arithmetic unit condition register; and

a plurality of general purpose flags (ACFs) that contain reduced condition information that is used for branching or conditional execution.

2. The machine of claim 1 wherein said plurality of ACFs have a format in which a bit or bits store the condition information from a number of execution units operating in parallel.

3. The machine of claim 1 further comprising an instruction memory for storing instructions that are executed conditionally, execution of said instructions not affecting the ACFs.

4. The machine of claim 1 wherein the SIMD machine operates on packed data instructions and one ACF is affected for each packed data operation.

5. The apparatus of claim 1 in which multiple PEs are employed and different PEs select different units to affect the ACFs.

6. A method of supporting conditional execution in a very long instruction word (VLIW) based array processor with subword execution, the method comprising:

providing general purpose flag bits (ACFs) that contain reduced condition information that is used for branching or conditional execution; and

specifying and setting a condition in ACFs based upon a condition code specification encoded in an instruction generating a condition.

7. The method of claim 6 wherein instructions that execute conditionally do not affect the ACFs.

8. The method of claim 6 wherein instructions that affect the ACFs execute unconditionally.

9. The method of claim 6 further comprising the steps of:

executing a packed data instruction where the execution of each sub-word of the packed data operation is dependent upon the associated subword ACF.

10. A hierarchical conditional execution instruction format comprising:

a plurality of instruction bits defining an instruction;

a 3-bit, 2-bit or 1-bit subset of said plurality of instruction bits providing an opcode extension encoding for each instruction that supports conditional execution;

a set of general-purpose arithmetic condition flags (ACFs) that store specified results from instruction execution; and

a set of arithmetic scalar flags (ASFs).

11. The hierarchical conditional execution instruction format of claim 10 wherein said 1-bit opcode extension is employed and said instruction conditionally executes on a true AFC condition if said 1-bit is a one, or said instruction conditionally executes on a false AFC condition if said 1-bit is a zero without affecting the AFCs.

12. The hierarchical conditional execution instruction format of claim 10 wherein said 2-bit opcode extension is employed and said instruction unconditionally executes if both bits are true and the ACFs are affected as defined by a SetCC instruction.

13. The hierarchical conditional execution format of claim 12 wherein the SetCC instruction includes a first opcode encoding field which specifies to which execution unit a specified condition applies.

14. The hierarchical conditional execution format of claim 13 wherein the SetCC instruction includes a second opcode encoding field which specifies a plurality of test conditions.

15. The hierarchical conditional execution format of claim 13 wherein each execution unit has a different specified condition or if an ALL encoding of the SetCC instruction is chosen then all the execution units use the same specified condition.

16. The hierarchical conditional execution format of claim 14 wherein a test condition can be specified to detect whether an overflow occurred on any data operation within a packed data execution.

17. The hierarchical conditional execution instruction format of claim 10 wherein said 2-bit opcode extension is employed and said instruction conditionally executes dependent upon the state of an ACF if either one of said bits is true and the other one of said bits is false, without affecting the ACFs.

18. The hierarchical conditional execution format of claim 10 wherein said 2-bit opcode extension is employed and said instruction unconditionally executes if both bits are false, without affecting the ACFs.

19. The hierarchical conditional execution instruction format of claim 10 wherein said 3-bit opcode extension is employed and said instruction specifies how the instruction is to be executed unconditionally or conditionally based on the ACFs and how the ACFs are affected utilizing the bits of said 3-bit opcode extension.

20. The hierarchical conditional execution instruction format of claim 15 wherein the set of ASFs flags represent the side effect from the instruction that is executing and wherein if two or more bits of said 3-bit opcode extension are true the AFCs are affected based on one of the conditions of said set of ASFs.

21. The hierarchical conditional execution format of claim 20 wherein said ASFs comprise carry (C), overflow (V), sign (N) and zero (Z) flags.

22. The hierarchical conditional execution instruction format of claim 10 wherein said 3-bit opcode extension is employed to specify an operation to be performed on one or more data elements of a packed data instruction and said instruction unconditionally executes the operation specified by the instruction on all said data elements without affecting the ACFs if all bits in said 3-bit opcode extension are false.

23. The hierarchical conditional execution instruction format of claim 10 wherein said 3-bit opcode extension is employed to specify an operation on one or more data elements of a packed data instruction said instruction conditionally executing based on the state of an AFC operation specified by the instruction on all said data elements or the operation does not occur at all.

24. The hierarchical conditional execution instruction format of claim 10 wherein said 3-bit opcode extension is employed to specify an operation on one or more data elements of a packed data instruction and said instruction conditionally executes only the data elements having a corresponding ACF flag of appropriate value for the specified true or false coding of said 3-bit opcode extension.

25. A method of condition generation comprising the steps of:

defining a set of arithmetic condition flags (ACFs);

determining side effects of a plurality of scalar conditions on an instruction by instruction basis;

setting a set of arithmetic scalar flags (ASFs) to save the determined side effects;

specifying a condition code utilizing a compare instruction; and

updating the ACFs based upon the specified condition code.

26. The method of claim 25 further comprising the step of:

combining a previous state of the ACFs with the result of a condition code test specified by a current compare instruction to create a complex condition.

27. The method of claim 25 wherein the condition code specifies a condition such as greater than (GT), less than (LT), equal (EQ) or less than or equal (LEQ).

28. The method of claim 27 wherein the compare instruction is further utilized to specify the desired conditions to be tested and two source registers to be compared.

29. The method of claim 28 wherein the compare instruction is further utilized to specify a data type covering packed data forms.

30. The method of claim 28 wherein the compare instruction is further utilized to specify a Boolean combination specification field.

31. The method of claim 26 further comprising the step of controlling branching in a sequence processor (SP) based upon the created complex condition.

32. The method of claim 26 further comprising the step of conditionally executing in a sequence processor (SP) and at least one processing element (PE) based on the created couplex consisting of a Boolean combination of multiple conditions. based upon the created complex condition.

33. The method of claim 26 further comprising the step of conditionally executing on a combination of multiple conditions based upon the created complex condition.

34. A system for generating complex conditions formed by a Boolean combination of relations comprising:

an arithmetic unit which receives at least two operands from a register file;

instruction control lines derived from a registered instruction in a processor pipeline, the instruction control lines including conditional execution control lines to control conditional operation as specified in an instruction;

the arithmetic unit producing a result and a latched arithmetic scalar condition state;

a first latch for holding the arithmetic scalar condition state for the instruction after the instruction has finished its execution state;

a second latch connected to the conditional execution control lines for holding instruction control signals for the instruction after the instruction has finished its execution state;

an arithmetic condition flag (ACF) generation unit for providing a Boolean combination of a present selected state with a previous state; and

an ACF latch for storing the previous state and feeding the previous state back to the ACF generation unit.

35. The system of claim 34 wherein the ACF latch is a programmer visible latch.

36. The system of claim 34 further comprising a multiplexer connected to receive said Boolean combination from the ACF generation unit and to controllably switch said Boolean combination or said AFC latch to branch logic in a sequence processor (SP).

37. The system of claim 34 further comprising an arithmetic scalar flag (ASF) latch switchably connected to the said first latch arithmetic scalar condition state output of the arithmetic unit.

38. The system of claim 37 wherein the switchable connection of the said first latch arithmetic scalar condition state output of the arithmetic unit and the ASF latch comprises a controllable multiplexer.

39. The system of claim 38 wherein an output of the controllable multiplexer controllably switches the arithmetic scalar condition state or the ASF latch output to the branch logic in a sequence processor (SP).

40. The system of claim 37 wherein the ASF latch is a programmer visible latch.

41. A single instruction multiple data stream (SIMD) machine with a controller (SP) and at least two processing elements (PEs), each PE in said SIMD machine comprising:

an arithmetic unit which receives at least two operands from a register file;

instruction control lines derived from a registered instruction that was received from the SP in a processor pipeline, the instruction control lines including conditional execution control lines to control conditional operation as specified in an instruction;

the arithmetic unit producing a result and a latched arithmetic scalar condition state;

a first latch for holding the arithmetic scalar condition state for the instruction after the instruction has finished its execution state;

a second latch connected to the conditional execution control lines for holding instruction control signals for the instruction after the instruction has finished its execution state;

an arithmetic condition flag (ACF) generation unit for providing a Boolean combination of a present selected state with a previous state; and

an ACF latch for storing the previous state and feeding the previous state back to the ACF generation unit.

42. The SIMD machine of claim 41 wherein the ACF latch of each PE is a programmer visible latch.

43. The SIMD machine of claim 41 wherein each PE further comprises a multiplexer connected to receive said Boolean combination from the ACF generation unit and to controllably switch said Boolean combination or said ACF latch to branch logic in a sequence processor (SP).

44. The SIMD machine of claim 41 wherein each PE further comprises an arithmetic scalar flag (ASF) latch switchably connected to said first latch arithmetic scalar condition state output of the arithmetic unit.

45. The SIMD machine of claim 44 wherein the switchable connection of said first latch arithmetic scalar condition state output of the arithmetic unit and the ASF latch of each PE comprises a controllable multiplexer.

46. The SIMD machine of claim 45 wherein an output of the controllable multiplexer of each PE controllably switches the arithmetic scalar condition state or the ASF latch output to the branch logic in a sequence processor (SP).

47. The SIMD machine of claim 41 wherein the arithmetic unit of each PE is one of a set of execution units comprising a multiply accumulate unit (MAU), an arithmetic logic unit (ALU), and a data select unit (DSU) each having an associated arithmetic condition flag (ACF) generation unit.

48. The SIMD machine of claim 47 wherein outputs from the ACF generation units for the MAU, ALU, DSU, and the ACF latch are controllably switched by a multiplexer to branch logic in a sequence processor.

49. An indirect very long instruction word (VLIW) processing system comprising:

a first processing element (PE) having a VLIW instruction memory (VIM) for storing instructions in slots within a VIM memory locations;

a first register for storing a function instruction having a plurality of group bits defining instruction type and a plurality of unit field bits defining execution unit type;

a predecoder for decoding the plurality of group bits and the plurality of unit field bits; and

a load mechanism for loading the function instruction in an appropriate one of said slots in VIM based upon said decoding, the first processor further comprising:

at least two execution units, each execution unit receiving at least two operands from a register file;

each execution unit having instruction control lines derived from a registered instruction in a processor pipeline, the instruction control lines including conditional execution control lines to control conditional operation as specified in an instruction to be executed;

each execution unit producing a result and a latched arithmetic scalar condition state;

each execution unit having a first latch for holding the arithmetic scalar condition state for the instruction after the instruction has finished its execution state;

each execution unit having a second latch connected to the conditional execution control lines for holding instruction control signals for the instruction after the instruction has finished its execution state;

each execution unit having an arithmetic condition flag (ACF) generation unit for providing a Boolean combination of a present selected state with a previous state; and

a single ACF latch for all of the execution units for storing the previous state and feeding the previous state back to the respective ACF generation unit.

50. The system of claim 49 wherein the ACF latch is a programmer visible latch.

51. The system of claim 49 wherein the PE further comprises a multiplexer connected to receive said Boolean combination from each of the ACF generation units and to controllably switch said Boolean combinations to branch logic in a sequence processor (SP).

52. The system of claim 49 wherein the PE further comprises an arithmetic scalar flag (ASF) latch switchably connected to the output of each of the execution units.

53. A system for generating complex conditions comprising:

an arithmetic unit which receives at least two operands from a register file;

instruction control lines derived from a registered instruction in a processor pipeline, the instruction control lines including conditional execution control lines to control conditional operation as specified in an instruction;

the arithmetic unit producing a result and a latched arithmetic scalar condition state;

a first latch for holding the arithmetic scalar condition state for the instruction after the instruction has finished its execution state;

a second latch connected to the conditional execution control lines for holding instruction control signals for the instruction after the instruction has finished its execution state;

an arithmetic condition flag (ACF) generation unit for providing a present selected state of a plurality of arithmetic condition flags (ACFs); and

an ACF latch for storing a previous state for the ACFs and feeding the previous state back to the ACF generation unit.

54. A single instruction multiple data stream (SIMD) machine with a controller (SP) and at least two processing elements (PEs), each PE in said SIMD machine comprising:

an arithmetic unit which receives at least two operands from a register file;

instruction control lines derived from a registered instruction that was received from the SP in a processor pipeline, the instruction control lines including conditional execution control lines to control conditional operation as specified in an instruction;

the arithmetic unit producing a result and a latched arithmetic scalar condition state;

a first latch for holding the arithmetic scalar condition state for the instruction after the instruction has finished its execution state;

a second latch connected to the conditional execution control lines for holding instruction control signals for the instruction after the instruction has finished its execution state;

an arithmetic condition flag (ACF) generation unit for providing a present selected state of a plurality of arithmetic condition flags (ACFs); and

an ACF latch for storing a previous state for the ACFs and feeding the previous state back to the ACF generation unit.

55. An indirect very long instruction word (VLIW) processing system comprising:

a first processing element (PE) having a VLIW instruction memory (VIM) for storing instructions in slots within a VIM memory locations;

a first register for storing a function instruction having a plurality of group bits defining instruction type and a plurality of unit field bits defining execution unit type;

a predecoder for decoding the plurality of group bits and the plurality of unit field bits; and

a load mechanism for loading the function instruction in an appropriate one of said slots in VIM based upon said decoding, the first processor further comprising:

at least two execution units, each execution unit receiving at least two operands from a register file;

each execution unit having instruction control lines derived from a registered instruction in a processor pipeline, the instruction control lines including conditional execution control lines to control conditional operation as specified in an instruction to be executed;

each execution unit producing a result and a latched arithmetic scalar condition state;

each execution unit having a first latch for holding the arithmetic scalar condition state for the instruction after the instruction has finished its execution state;

each execution unit having a second latch connected to the conditional execution control lines for holding instruction control signals for the instruction after the instruction has finished its execution state;

each execution unit having an arithmetic condition flag (ACF) generation unit for providing a present selected state of plurality of arithmetic condition flags (ACFs); and

a single ACF latch for all of the execution units for storing a previous state for the ACFs and feeding the previous state back to the respective ACF generation unit.